SOFTHEQUE ORDINATEUR

6

Avec la cassette de vidéo-jeux et programmes pour ZX SPECTRUM et VIC 20

LANGAGE MACHINE:
• 2e PARTIE

PSEUDOCODE:
• LE BESTIAIRE

LE BASIC ET LES PERIPHERIQUES



Promopublications

M 6111-6 - 85 F Belgique: 680 FB Suisse: 28 FS Canada: 14$ C

SOFTHEQUE N. 6

MENSUEL - SEPTEMBRE 1985

Directeur: Franco Bozzesi

Ont collaboré:
MICHELLE BLEIN
GEORGES RIEBEN
MAX CELLINI
ROBERTO TREPPIEDI
DIDIER DUCHESNE
ALDO CAMPANOZZI
GEORGETTE LHOPITAL
ALEX VALLONE
HELENE RACCAH
ALBERTO BARBATI
YOLANDE TERISSE
ANTONIO LUCARELLA
CATHERINE JUERY
DANIELE RIEFOLI
JEAN CAPOBIANCO

SOFTHEQUE est une création de PROMOPUBLICATIONS, Sarl au capital de 20.000 F. 312179195 B.R.C. Paris.
Rédaction, administration, vente, publicité, siège social: 34, Champs-Elysées, 75008 Paris. Tel. (1) 5634850
Distribué en France par: N.M.P.P.
Imprimerie: ALIGRAF Milan - Italie
Directeur de la publication: Franco Bozzesi
Numéro de commission paritaire: en cours.
Dépôt légal n. 50579 du 8 Juin 1984.

LE "BESTIAIRE"

Le lecteur attentif qui a suivi notre série d'articles sur le pseudocode aura certainement remarqué que l'essentiel du sujet s'est de plus en plus déplacé vers la STRUCTURE DES DONNEES.
Ceci n'est pas dû au hasard et nous vous en expliquons tout de suite la raison.
L'ALGORITHME exprime la logique suivie de la solution proposée par le programmeur en mettant en évidence les choix, les répétitions et les transformation élémentaires (instructions) nécessaires pour atteindre le but final.
LA STRUCTURE DES DONNEES représente au contraire le choix du programmeur pour découvrir une structure interne dans les données dont dispose le programme. Cette structure interne est tellement importante qu'à partir de là on pourra deviner l'algorithme.
Lorsque, dans une des organisations simples ou complexes, nous résumons la structure des données, cela signifie que nous commençons à en apercevoir la solution.
Certaines des méthodes les plus connues font de cette intuition la partie principale de leur enseignement: en établissant un schéma des données on établit aussi un système de résolution.
Sans vouloir trop s'avancer, il est vrai qu'une solution est déjà cachée dans les données que nous possédons: il s'agit de l'en extraire, comme le sculpteur qui, d'une motte d'argile, extraira une figure.
C'est pourquoi, la possibilité d'organiser les données de façon à en exprimer les rapports internes est un des plus puissants instruments dont dispose le programmeur.
Dans cet article, nous approfondirons notre connaissance sur les structures de données complexes, en introduisant le concept de "lien" entre les listes.

LIENS ET POINTEURS

Un pointeur est une (quelconque) variable numérique qui contient l'INDICE d'une structure de données simples (vecteur ou matrice). Par exemple, dans une liste, le pointeur au premier élément libre de la liste est une variable numérique qui contient l'adresse du premier élément vide du vecteur LIST (voir article 4).
Jusqu'à présent les pointeurs étaient externes aux structures données complexes que nous avons étudiées: le pointeur de



début et de fin de liste circulaire, le pointeur à l'élément de tête d'une queue et ainsi de suite. Ainsi nos structures complexes pourraient être représentées comme d'après la figure 1 où, pour représenter le pointeur, nous avons utilisé une case avec une flèche qui indique l'élément pointé (dans la réalité la flèche est remplacée par l'indice contenu dans P).

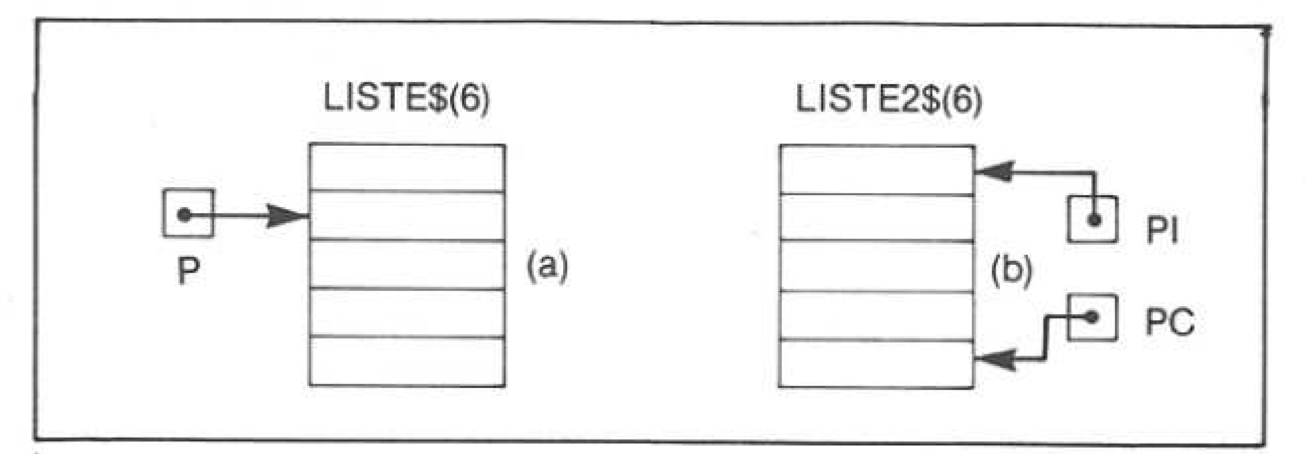


Fig. 1. Exemple de représentation graphique d'une liste (a), et d'une liste circulaire (b).

Les structures des données complexes ainsi décrites sont dites LINEAIRES, car les éléments se trouvent "les uns derrière les autres" placés dans les vecteurs de support et il n'est donc pas nécessaire d'ajouter des informations pour spécifier où se trouve l'élément suivant. En effet, au cours de l'édition d'une liste ou d'une queue, il suffit de connaître le début et la fin des données et imprimer tous les éléments inclus.

Avec les structures linéaires, le programme a la possibilité d'exprimer des liens linéaires de données: les cartes contenues dans un jeu, la queue au guichet d'un bureau, la liste des rendez-vous, l'annuaire téléphonique personnel (avec adjonctions, radiations et modifications).

Mais si, au contraire, nous voulons représenter des liens entre données groupées dans une seule structure nous devons exprimer un RAPPORT entre les données, qui ne soit pas simplement la succession.

Voici un exemple concret et en même temps créatif pour nos lecteurs.

Actuellement nous apprenons le nom de différents animaux dans une langue étrangère, par exemple l'anglais.

Nous aimerions les mémoriser dans une liste sur notre fidèle OP et en même temps les garder sous contrôle de façon à pouvoir obtenir des listes caractéristiques telles que:

- Liste de tous les mammifères (reptiles, poissons, oiseaux ou amphibies).
- Liste des "familles": le mâle, la femelle et le petit (par exemple: taureau, vache et veau).

Notre idée est, avec le temps, d'agrandir notre "bestiaire", en ajoutant tous les jours quelque autre nom, de façon à l'accrocher aux autres, sans prétendre introduire tous les noms en même temps.

Chaque animal est donc un élément séparé et aussi une information qui s'encastre dans la mosaïque du bestiaire déjà introduit.

D'autre part, il peut être très utile de construire de tels dictionnaires personnels, tant pour les animaux que pour n'importe quel autre terme de langage (verbes, objets, etc.). Cela deviendra une façon amusante (et imprévisible) d'enrichir son vocabulaire anglais.

Dans notre exemple, les seules informations de liens entre les animaux seront la CLASSE (mammifères, amphibies, reptiles, poissons ou oiseaux) et les SOUS-GROUPES (mâle, femelle, petit). Cependant, avec le même système nous pourrons ajouter des classifications plus subtiles en multipliant ce schéma pour tous les nouveaux attributs désirés.

Nous vous conseillons de commencer avec un maximum de sept: avec 15 notre bestiaire se transformerait déjà en un beau jardin zoologique!

Pour revenir à notre problème technique, nous voyons que nous avons touché le point crucial qui différencie une ligne linéaire d'une liste avec des liens: nous voulons mémoriser non seulement les données mais aussi les rapports qui les unissent. Dans notre exemple, lorsque nous introduisons le terme VACHE, nous voulons qu'il ait un lien avec TAUREAU et avec VEAU, de façon à pouvoir, par la suite, reconstruire la famille.

Les liens entre les données sont mémorisés sous forme de pointeurs aux autres éléments de la structure. Ces pointeurs, à la différence de ceux des listes linéaires, sont à considérer comme partie intégrante de la structure des données complexes.

Le nom de cette structure est LISTE LIEE du fait que les différents éléments sont pour ainsi dire "enchaînés" par les pointeurs (en anglais "LINKED LIST").

Un élément d'une liste liée comprend 2 composants:

a. L'information, qui contient la donnée mémorisée avec toutes les informations importantes.
b. L'ensemble des pointeurs de liens, un pour chaque type de lien que l'on entend établir.

Cette liste étant une structure de données complexes, pour la réaliser il faut faire appel à des structures de données élémentaires.



Composants élémentaires:
a. L'information est contenue dans un vecteur (variable avec indice) appelé INFO$ (m)
b. Les pointeurs de liens: puisqu'il y en a une série pour chaque élément INFO$, il s'ensuit que ceux-ci sont des variables à doubles indices: POI(i,m). Le premier indice indique le genre de lien et le second indique l'élément auquel il se rapporte.

En ce qui concerne notre bestiaire anglais:
a. L'information relative au nom de l'animal (que nous introduisons, avec sa traduction, dans le vecteur INFO$).
b. Les liens seront: CLASSE et SUCCESSEUR DE FAMILLE.

Mais que mémorise-t-on exactement pour chaque pointeur de lien? Voici la règle magique du lien:
Chaque pointeur de lien pointe à l'élément de la liste liée qui est son SUCCESSEUR DANS L'ORDRE DE LIEN.
Et donc, dans notre cas, nous aurons pour n'importe quel élément K:
POI (1,K): pointe à l'animal suivant dans la liste de la même CLASSE: (au mammifère suivant si INFO(K) est un mammifère, au poisson suivant si INFO (K) est un poisson, et ainsi de suite).
POI (2,K): pointe l'animal dans la liste liée de "parenté" (par exemple si INFO (K) est "CHAT", pointe à "CHATTE" ou à "CHATON", selon l'animal qui a été introduit).
Extérieurement à la liste liée, les pointeurs servent à chaque début de lien: il faut le pointeur au premier mammifère, au premier poisson, au premier oiseau et au premier amphibie de la liste. (Dans notre cas un pointeur de début de la première famille n'est pas nécessaire, aucune d'elles n'étant liée aux autres).

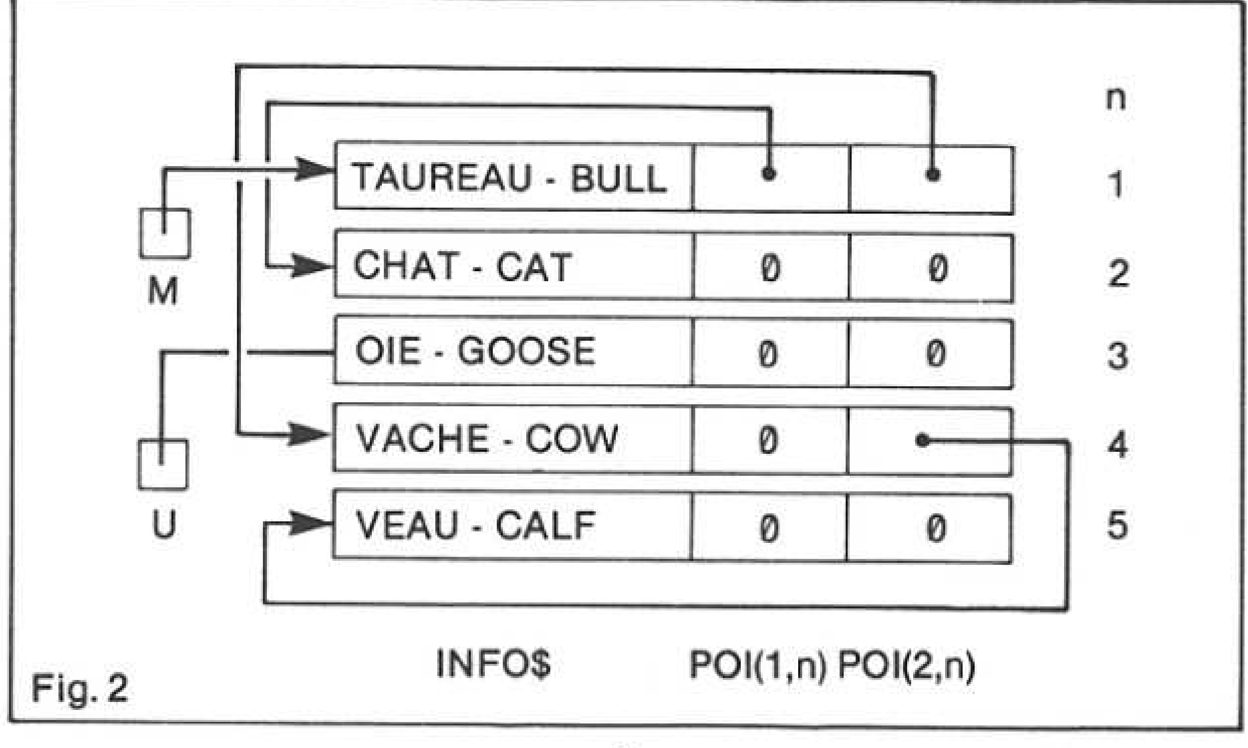


Fig. 2

Une dernière chose à savoir, avant d'examiner les algorithmes de gestion de la liste, est que nous utiliserons la valeur conventionnelle 0 (zéro), pour indiquer le pointeur vide, c'est-à-dire la fin d'une chaîne de liens.
Pour recenser tous les mammifères par exemple, nous partirons du premier (pointé par le pointeur de début) et nous suivrons les pointeurs de liens POI(1) jusqu'à trouver le pointeur vide.
Donnons à présent un exemple graphique d'une liste dans laquelle nous avons introduit 5 animaux dans l'ordre suivant: TAUREAU, CHAT, OIE, VACHE, VEAU.
Sur le dessin, M et U sont les pointeurs externes au premier mammifère et au premier oiseau de la liste. Comme vous voyez, la liste contient les animaux dans l'ordre où ils ont été introduits, mais en suivant les flèches, nous pouvons reconstruire les séries des mammifères (en suivant POI (1,)) et la famille de TAUREAU (en suivant POI (2,)).
Sur cet autre dessin, nous remplaçons les flèches par les nombres indice d'élément et nous aurons une idée exacte du contenu de nos variables:

1
M

3
U

TAUREAU - BULL	2	4	1
CHAT - CAT	0	0	2
OIE - GOOSE	0	0	3
VACHE - COW	0	5	4
VEAU - CALF	0	0	5

Fig. 3

LA CONSTRUCTION DE LA LISTE LIEE

Réfléchissons à présent sur la manière d'augmenter notre liste liée à partir de zéro, en introduisant un animal à la fois.
Tout d'abord, nous avons besoin d'un programme pour l'acquisition des données. Nous en reportons un figure 4

```
10 DIM INFO$(100):DIM POI(2,100):DIM MOPRA(5)
30 INPUT "INSERTION (1),EXTRACTION (2),FIN (3) ?";CT
40 IF CT=1 THEN GOTO 50
41 IF CT=2 THEN GOTO 570
42 IF CT=3 THEN GOTO 700
50 PRINT"      PHASE D'INTRODUCTION DES DONNES"
```




```
51 IF CT=2 THEN GOTO 570
60 "NOM FRANCAIS DE L'ANIMAL"
70 IF NI$="STOP" THEN GOTO 30
80 INPUT "NOM ANGLAIS DE L'ANIMAL ?",EN$
90 INPUT "MALE (M),FEMELLE (F) O PETIT (P) ?";SE$
100 INPUT "MAMMIFERE(1),OISEAU(2),POISSON(3),      REPTILE(4),AMPHIBIE(5) ?";CL
110 INPUT "ANIMALE PARENTE ?";AP$
120 LET B$=" ":LET L1=8-LEN(NI$)
130 LET L2=8-LEN(EN$)
140 FOR A=1 TO L1:S1$=S1$+CHR$(32):NEXT A
150 FOR A=1 TO L1:S2$=S2$+CHR$(32):NEXT A
160 LET INF$=NI$+S1$+S2$+EN$+SE$
```

Fig. 4. Recueil des données

Résumons les informations que le programme demande:
- —nom français de l'animal: NI$
- —nom anglais de l'animal: EN$
- —mâle, femelle ou petit: SE$
- —classe (mammifère, oiseau, poisson, reptile ou amphibie): CL
- —animal de la même famille: AP$

Nous remarquons que les informations NI$, EN$ et SE$ sont liées dans la variable INF$ (instruction 160): le signe + est l'opérateur de l'enchaînement alphanumérique. Les instructions 120-150 servent à uniformiser à 8 caractères le nom français et anglais, de façon à ce qu'ils occupent une position fixe dans INF$.

```
170 REM ********************************
180 GOSUB 300:REM ** INTRODUISEZ L'ANIMAL *
185 IF LEN(AP$)=0 THEN GOSUB 370:REM ********* MISE A JOUR LIEN DE CLASSE *****
190 IF LEN(AP$)<>0 THEN GOSUB 470:REM ******** MISE A JOUR LIEN DE PARENTE **
200 IF FOUND=0 THEN GOTO 270
210 REM ********************************
220 PRINT "* ANIMAL ";NI$;" INTRODUIT *":PRINT
240 LET N=N+1:IF N<10 THEN GOTO 60
260 REM ********************************
270 PRINT"* ANIMAL PARENT NON TROUVE *"
275 GOSUB 370:REM *** MISE A JOUR LIEN DE CLASSE ***
280 GOTO 210
```

Fig. 5. Rappel des fonctions d'insertion et mise à jour des liens.

Figure 5 nous reportons la suite du programme, qui prévoit d'introduire l'animal de la liste. Observez: tandis que l'insertion d'un animal est fixe, la mise à jour du lien de parenté n'est exécutée que lorsqu'il n'y a pas d'animal de la même famille!
En effet, dans notre liste, les animaux liés par familles ne sont pas liés par classes (voir la figure 2: VACHE ne peut être atteinte

qu'à travers TAUREAU en suivant le lien de parenté, c'est-à-dire POI (2,1)).
Les instructions 270-280 servent à mettre à jour le lien de classe lorsque l'animal parent n'est pas présent dans la liste, c'est-à-dire FOUND = O. Dans ce cas il s'agit d'une erreur et il faut empêcher que l'animal devienne inaccessible.
A présent nous sommes capables d'écrire l'algorithme pour l'introduction de l'animal. Il faut avant tout penser à une variable LL qui indique la dernière adresse occupée de la liste: au départ LL vaudra zéro.
Nous avons donc:

```
DEBUT Introduisez l'animal
  LL = LL + 1
  SI LL > FDONNEES ALORS STOP
  INFO$(LL) = INF$
  POI(1,LL) = O;POI(2,LL) = O
FIN
```

L'insertion est simple: on attribue à INF$ sa place dans la liste en ajoutant 1 à LL, et en vérifiant que LL ne soit pas supérieure à la longueur totale de INFO$, c'est-à-dire FLISTE.
Considérons à présent la mise à jour du lien de classe: il s'agit d'examiner toute la série des animaux de la même classe, trouver le dernier (qui aura 0 comme POI(1,n)) et modifier POI(1,n) de façon à pointer à l'élément à peine introduit, c'est-à-dire le contenu de LL.
Les cinq pointeurs de classes (Mammifères, Oiseaux, Poissons etc.) sont dans un vecteur MOPRA de 5 éléments; la variable CL nous indique lequel utiliser pour trouver le début de chaque classe.

```
DEBUT. Mise à jour lien de classe.
  SI PI = O ALORS MOPRA (CL) = LL
  AUTREMENT
    REPETEZ
    JUSQU'A POI(1,PI) # O
      PI = POI(1,PI)
    FIN_REPETEZ
  POI(1,PL) = LL
  FIN_SI
FIN
```

Le "SI" sert à contrôler si nous n'avons pas encore introduit d'animal de cette classe: dans ce cas on initialise le pointeur MOPRA approprié.
Autrement on SUIT les "flèches": on parcourt la liste liée-(cycle REPETEZ) jusqu'à localiser l'élément dont le pointeur POI(1,n) vaut 0. C'est alors que nous mettons à jour ce pointeur avec le contenu de LL, c'est-à-dire le pointeur au nouvel élément à peine



ajouté. Le programme se trouve fig. 6.
Etudions le procédé pour l'introduction d'animaux: TAUREAU, CHAT, VACHE, en ce qui concerne le lien de classe.
VACHE a aussi un lien de parenté avec TAUREAU: nous devons donc mettre à jour ce lien de parenté, c'est-à-dire POI (2,1). Afin de trouver TAUREAU dans la liste nous pourrons: 1) examiner INFO$ du début jusqu'à LL; 2) suivre le lien de mammifère du début jusqu'au pointeur Ø.
Dans le premier cas nous prendrons en considération le vecteur INFO$ comme si c'était une structure élémentaire, et nous examinerons tous les animaux, sans distinction de classe. Dans le deuxième cas, nous examinerons la structure tout au long du parcours de la liste liée, et ensuite uniquement les animaux de la même classe.
Les deux solutions sont possibles, avec une préférence pour la solution 2 qui empêche de mettre dans la même famille deux animaux de classes différentes: nous éviterons ainsi de mettre SERPENT et CHEVAL dans la même famille!
De toute façon la solution n° 1 est en tout et pour tout identique à la répétition du programme: "Mise à jour du lien de classe" et donc nous laissons au lecteur le soin de la réaliser pour lui-même.
Par contre, nous reportons ici la solution n° 1, de façon à mettre en évidence le fait que INFO$ peut être vue également comme structure indépendante et gérée comme un simple vecteur:

```
DEBUT. MISE A JOUR LIEN DE PARENTE
 J = 1
 REPETEZ
 JUSQU'A(J < LL) AND (AP$ # INFO$(J))
  J = J + 1
 FIN_REPETEZ
 SIN INFO$(J) = AP$
 ALORS
  SI POI(2,J) # O
  ALORS
   J = POI(2,J)
  FIN_SI
  POI(2,J) = LL
 AUTREMENT
  FOUND = Ø"animal non trouvé"
FIN_SI
```

Remarquez que dans le programme, après avoir trouvé l'animal de la même famille, il faut contrôler si son pointeur de parenté POI(2,J) est déjà établi. Dans ce cas on parcourt le lien d'un élément et on ajourne ensuite le lien de parenté. Figure 6, le programme BASIC traduit cet algorithme.

```
290 REM *********************************
300 REM ******* INTRODUCTION ANIMAL
310 LET LL=LL+1
320 IF LL>FDATI THEN GOTO 780
330 LET INFO$(LL)=INF$
340 LET POI (1,LL)=0:LET POI (2,LL)=0
350 RETURN
360 REM ***********************************
370 REM *** MISE A JOUR LIEN DE CLASSE ***
380 LET PI=MOPRA(CL)
390 IF PI=0 THEN LET MUPRA(CL)=LL:GOTO 450
400 IF POI (1,P)=0 THEN GOTO 440
410 LET PI=POI(1.PI)
420 GOTO 410
430 LET POI(1,PI)=LL
440 RETURN
460 REM *********************************
470 REM *** MISE A JOUR LIEN DE PARENTE **
480 LET L3=LEN(AP$)
500 IF AP$=INFO$(J.1 TO L3) THEN 520
510 NEXT J
520 IF INFO$(J,1 TO L3)<>AP$ THEN LET FOUND=0:RETURN
530 IF POI (2,J)<>0 THEN LET J=POI (2,J)
540 LET POI (2,J)=LL:LET FOUND=1
550 RETURN
```

Fig. 6. Insertion et mise à jour des liens.

LA PHASE D'EXTRACTION

Jusqu'à présent nous avons vu comment gérer la partie d'introduction de données dans la liste liée. Celle-ci s'appelle phase de "UPDATE" c'est-à-dire (mise à jour).
Il nous faut maintenant réaliser les algorithmes qui nous permettent de relire les informations. Cette partie s'appelle phase de "INQUIRY" (en français "interrogation") et en général les programmes qui, comme le nôtre, rangent et interrogent les données, s'appellent programmes de "Inquiry-Update".
L'interrogation est une fonction importante pour obtenir le maximum des informations mémorisées: plus on a de liberté pour faire des "inquiries" complexes et majeure est l'impression de puissance que notre programme nous transmet.
Une "inquiry" se divise en deux parties: un critère de choix des éléments à examiner, et le genre d'opération que l'on veut exécuter. Par exemple, sur la liste, dans un revue spécialisée sur les moteurs de voitures, l'interrogation:
"Quelles sont les voitures PEUGEOT offertes?" on peut distinguer entre:
1. Le choix des seules voitures PEUGEOT
2. La liste de tous les modèles.



Dans les programmes plus sophistiqués, le critère de choix peut être exprimé d'une manière très complexe, et les opérations peuvent être différentes (par ex.: mettre en ordre, dresser une liste, additionner, etc.).
Pour revenir à notre exemple, nous nous limitons à donner la possibilité de choisir l'une des cinq classes d'animaux: la fonction est toujours celle d'énumérer les animaux introduits en mettant par contre en évidence l'ordre de parenté.
Afin de parcourir la liste selon l'ordre de parenté, mais en choisissant tous les animaux d'une même classe, il faut gérer exprès POI(1,n) et POI(2,n). Il s'agit de descendre le long de POI(1,n) jusqu'à trouver un POI(2,n) différent de 0. Alors nous parcourons le lien de parenté jusqu'à épuisement. Nous recommençons ensuite à suivre POI(1,n). L'algorithme se présente donc comme deux cycles boucles "REPETEZ JUSQU'A":

```
INIZIO. Balayage liste suivant les liens
  PC = Premier pointeur de la liste de classe
  Imprimez "DEBUT liste"
  REPETEZ
  JUSQU'A PC # O
    Imprimez INFO$(PC)
    PF-POI(2,PC)"place le pointeur de parenté"
    REPETEZ
    JUSQU'A PF # O
      Imprimez INFO$(PF)
      PF = POI(2,PF)
    FIN__REPETEZ
    Imprimez "fin famille"
    PC = POI(1,PC)
  FIN__REPETEZ
  Imprimez "fin liste"
FIN
```

Figure 7 nous voyons la réalisation de cet algorithme.

```
560 REM ********************************* --
570 PRINT "          PHASE DE INQUIRY"
580 PRINT"SELECTION    "
590 INPUT "MAMMIFERE(1),OISEAU(2),POISSON(3),        REPTILE(4),AMPHIBIE(5) ?";CL
600 LET PC=MOPRA(CL)
605 PRINT"DEBUT LISTE"
610 IF PC=0 THEN GOTO 685
615 PRINT" ";INFO$(PC);
620 LET PF=POI(2,PC)
630 IF PF=0 THEN GOTO 671
640 PRINT"--";INFO$(PF);
650 LET PF=POI(2,PF)
670 GOTO 630
671 PRINT"**"
675 LET PC=POI(1,PC)
680 GOTO 610
```

```
685 PRINT" FIN DE LA LISTE "
688 REM ***********************************
690 GOTO 570
695 REM ***********************************
700 STOP

READY.
```

Fig. 7. Ainsi, nous parcourons la liste en suivant les différents liens.

```
NOM FRANCAIS DE L'ANIMAL? STOP
INTRODUCTION(1) OU EXTRACTION(2)
 OU FIN(3)? 2
PHASE DE INQUIRY
SELECTION
MAMMIFERE(1) OISEAU(2) POISSON(3
) REPTILE(4) AMPHIBIE(5)? 1
DEBUT DE LA LISTE
VACHE CAW F-- TAUREAU BULL M **
CHIEN DOG M-- CHIENNE BITCH F**
PHASE DE INQUIRY
SELECTION
MAMMIFERE(1) OISEAU(2) POISSON(3
) REPTILE(4) AMPHIBIE (5)? 2
DEBUT DE LA LISTE
OIE GOOSE F **
FIN DE LA LISTE
PHASE DE INQUIRY
SELECTION
MAMMIFERE(1) OISEAU(2) POISSON(3
) REPTILE(4) AMPHIBIE(5)? 4
DEBUT DE LA LISTE
VIPERE ADDER M **
FIN DE LA LISTE
PHASE DE INQUIRY
SELECTION
MAMMIFERE(1) OISEAU(2) POISSON(3
) REPTILE(4) AMPHIBIE (5)?
```

Fig. 8. Exemples de INQUIRIES.

Dans le prochain numéro, nous examinerons le concept de FILE uni à celui de LISTE LIEE afin d'obtenir la gestion d'un FILE à INDICE.



LE BASIC ET LES PERIPHERIQUES

Maintenant nous sommes arrivés au dernier article de notre cours de BASIC; après avoir examiné les principales instructions de ce simple langage, nous sommes prêts à décrire ce qui, probablement, suscite le plus grand intérêt parmi les utilisateurs: le graphisme.
Le Spectrum a été le premier ordinateur familial à disposer d'un graphisme à haute résolution, facilement utilisable avec de simples instructions BASIC.
Avant tout, voyons ce que l'on entend par haute et basse résolution. Normalement, la basse résolution est celle utilisée pour l'édition des inscriptions. Elle équivaut donc au nombre de positions-caractères disponibles sur l'écran qui, dans notre cas, correspondent à 32 x 22, c'est-à-dire 704. La haute résolution, au contraire, est utilisée pour des dessins ou graphiques d'une certaine complexité; elle correspond au nombre de points adressables un à un sur l'écran; chaque point s'appelle pixel, contraction des deux mots anglais "picture element".
Le Spectrum peut gérer l'image vidéo en haute résolution composée de 256 x 192 pixels, pour un total de 49 152 points. Comme on peut facilement le remarquer, on dispose de cette façon d'une résolution vraiment supérieure grâce à laquelle on peut réaliser des dessins de qualité.
Voyons à présent comment gérer les potentialités graphiques que nous offre notre Sinclair.
La première et aussi la plus simple des instructions à examiner est PLOT x,y avec laquelle nous pouvons "noircir" un point indiqué par les coordonnées x et y. En travaillant en haute résolution, les coordonnées sont relatives à l'angle du bas, à gauche, et donc en se référant au schéma reporté ensuite, le point au centre de l'écran correspond aux coordonnées 128,96.
Naturellement, en contrôlant cette instruction au moyen de cycles, nous pouvons aussi tracer des lignes, des courbes et des arcs, mais cela serait assez lent; c'est pourquoi le Spectrum met à disposition de l'utilisateur deux autres instructions très pratiques: DRAW et CIRCLE, respectivement pour le tracé de lignes (ou arcs) et de circonférences.
La syntaxe de l'instruction DRAW est:
DRAW X,Y,Z
où X indique la distance sur l'axe des abscisses du dernier point

dessiné, tandis que Y représente la distance des coordonnées sur l'axe.
Les valeurs peuvent être, naturellement, soit positives, soit négatives: elles seront négatives si le point à atteindre est plus bas ou à gauche du dernier point.
Par exemple si nous voulons dessiner un carré de 50 de côté, qui ait le sommet gauche aux coordonnées 100, 100, nous procèderons ainsi:

```
10 PLOT 100,100
20 DRAW 50,0
30 DRAW 0,50
40 DRAW -50,0
50 DRAW 0,50
```

La première ligne du programme fait apparaître un point de coordonnées 100,100 que nous avons établi comme étant le sommet gauche du carré et qui sert uniquement comme point de départ pour les lignes à tracer. La ligne 20 trace une ligne de 50 pixels à droite du point précédemment dessiné. La ligne 30 trace au contraire une ligne de 50 pixels en bas, à partir du dernier point qui n'est plus maintenant celui déterminé par l'instruction PLOT, mais le dernier point appartenant à la ligne tracée auparavant avec l'instruction DRAW de la ligne 20; les lignes 40 et 50 ferment le carré en traçant d'abord une ligne de 50 pixels à gauche (remarquez le moins) et ensuite une de 50 pixels vers le haut.
Précédemment, nous avons vu que l'instruction DRAW se composait en réalité de trois paramètres (bien que le troisième soit facultatif) et servait à tracer des arcs de cercle, en spécificant simplement l'angle autour duquel le segment doit tourner. Par exemple, si nous voulons dessiner un demi-cercle, d'un diamètre de 30 pixels, qui commence au point 20, 80, nous devrons utiliser un programme tel que:

```
10 PLOT 20,80
20 DRAW 0, -30,PI
```

où la ligne 10 établit le point de départ, tandis que la suivante trace une ligne inclinée vers le bas avec une courbe égale à PI (3.1415926) qui correspond à 1/2 radian.
Pour, au contraire, tracer des cercles entiers, nous pouvons recourir à l'instruction CIRCLE qui permet de dessiner directement des circonférences dans n'importe quelle position de l'écran. La syntaxe de l'instruction est:

```
CIRCLE X,Y,R
```

Dans ce cas, X et Y sont les coordonnées du point central tandis que R indique le rayon, exprimé naturellement en pixels. Ceci dit, si nous voulons dessiner un cercle d'un diamètre de 40



pixels, qui ait le centre du point 128,90 il suffira d'écrire l'instruction suivante (exprimée en mode direct):

```
CIRCLE 128,90,20
```

Vous verrez ainsi le cercle se développer sous vos yeux.
Jusqu'à présent nous avons vu comment procéder pour tracer des points, des lignes ou des cercles; si, par hasard, nous devions les effacer, nous pourrions utiliser la même instruction employée pour dessiner, en spécifiant toutefois la clause OVER.
Faisons un exemple pratique: traçons tout d'abord une ligne:

```
10 PLOT 40,10
20 DRAW 50,0
```

A présent, pour l'effacer, nous utiliserons un programme semblable, en introduisant l'instruction OVER 1, avec laquelle nous obtenons une "inversion" des couleurs (comme si nous repassions au corrector la ligne déjà tracée).

```
10 PLOT OVER 1;40,10
20 DRAW OVER 1;50,1
```

Remarquez qu'il faut aussi spécifier un nombre (0 ou 1) après OVER; en effet, 0 rend nul l'effet de l'instruction, tandis qu'avec 1 cette dernière est mise en fonction.
Maintenant il nous faut expliquer comment agit en réalité l'instruction OVER. Pratiquement, celle-ci superpose les points qui lui sont spécifiés à ceux déjà présents sur l'écran et le résultat est une inversion (effacement) des points qui coïncident. Naturellement, dans l'exemple précédent, puisque toutes les instructions sont égales, tous les points coïncident et on obtient le résultat d'invertir (d'effacer) le dessin. Cette instruction particulière peut être employée avec PRINT, bien que, dans ce cas, elle ne soit pas d'une grande utilité.
Examinons l'utilisation des couleurs à l'intérieur des instructions graphiques.
Le Spectrum dispose de 8 couleurs sélectionnables pour textes, fond et pourtour. Les instructions qui s'occupent de la gestion des couleurs sont: INK, PAPER et BORDER, suivies du code correspondant à la couleur que l'on entend utiliser. Faisons un exemple pratique:

```
10 BORDER 1
20 PAPER 0
30 INK 6
40 CLS
50 PRINT "ESSAYEZ LES COULEURS"
60 STOP
```

Les lignes 10, 20 et 30 attribuent respectivement le bleu pour le pourtour, le noir (0) pour le fond et le jaune (6) pour les caractères. L'instruction CLS est nécessaire lorsque l'on désire que les couleurs choisies agissent sur tout l'écran; dans le cas contraire, seul ce qui a été imprimé après les instructions d'attribu-

tion aura les couleurs désirées. Dans l'exemple, vous pouvez remarquer que tout l'écran, pourtour excepté, devient noir et la phase de la ligne 50 est imprimée en jaune.
Avec le Spectrum, cependant, on peut faire davantage: en effet, chaque caractère peut avoir une couleur et un fond différents, indépendamment des autres. Pour définir une couleur qui agisse uniquement sur une fonction de PRINT, sans influencer le reste, nous procèderons ainsi:

```
10 PRINT INK 2; "ESSAI"
20 PRINT PAPER 1;INK 7; "AVEC DIFFERENCES"
30 PRINT AT 10,8;INK 7; "COULEURS"
```

De cette façon, la première phrase sera imprimée en rouge, indépendamment des couleurs choisies auparavant; la seconde phrase sera, par contre, en blanc sur bleu, tandis que la troisième sera imprimée en vert aux coordonnées 10, 8. Notez que, pour utiliser les instructions de couleurs dans le seul secteur d'une PRINT, elles doivent suivre cette dernière et les séparer par ";" comme vous le constaterez avec l'exemple. Répétons que les couleurs ainsi obtenues valent seulement à l'intérieur d'une PRINT (ou autres instructions graphiques).
Le même discours vaut pour les instructions graphiques traitées précédemment; rappelez-vous toutefois que, pour chaque position de caractère, on ne peut avoir plus de 2 couleurs, c'est-à-dire une de fond et une d'écriture.
Cela dit, vous vous rendrez compte que lorsque deux lignes en haute résolution avec différentes couleurs se croisent, il en résulte une superposition très peu esthétique. Essayez ce qui suit:

```
10 PLOT INK A;10,30
20 DRAW INK 4;30,0
30 PLOT INK 2;20,5
40 DRAW INK 2;0,40
```

Vous remarquerez qu'au point d'intersection des deux lignes, nous avons une superposition de couleurs dans une position-caractère. Afin d'éviter ce genre d'inconvénients, nous vous conseillons d'utiliser la haute résolution avec une seule couleur lorsque vous devez faire des dessins compliqués avec des lignes qui s'entrecroisent.
Comme nous l'avons déjà dit, dans chaque position-caractère ne peuvent coexister que 2 couleurs, mais pour chaque secteur on peut choisir l'intensité de luminosité et on peut faire en sorte qu'il soit imprimé clignotant ou en champ inversé. Les instructions avec lesquelles vous pouvez obtenir ces effets sont: BRIGHT pour la luminosité, FLASH pour le clignotant et INVERSE pour l'impression à couleurs inversées.
Par l'intermédiaire de l'instruction BRIGHT suivie de 0 ou de 1, on peut spécifier si les textes devront être à luminosité normale



(0) ou extra (1). Cette instruction aussi, comme les précédentes, peut être utilisée individuellement ou à l'intérieur d'une instruction graphique. Dans ce cas, nous le répétons pour la énième fois, l'effet durera seulement dans les limites de cette instruction.
Pour obtenir des textes clignotants, utiles pour attirer l'attention, vous pourrez faire appel à l'instruction FLASH, elle aussi suivie de 0 ou de 1, selon que l'on veut obtenir ou non l'effet de clignotement. Si, par contre, il vous suffit d'obtenir des textes en mode inversé vous pouvez utiliser l'autre commande INVERSE, suivie, comme toujours, de 0 ou de 1. Pour ces deux instructions, ce que nous avons dit précédemment à propos de BRIGHT et des instructions de couleurs reste valable: toutes peuvent être employées indépendamment ou dans le cadre d'une PRINT, DRAW, etc.
Faisons un bref programme qui utilise en même temps toutes les instructions jusqu'ici citée:

```
 10 BORDER 1
 20 PAPER 1
 30 INK 7
 40 CLS
 50 PRINT AT 5,2;"SPECTRUM"
 60 PRINT AT 7,2;INK 4;"SPECTRUM"
 70 PRINT AT 9,2;BRIGHT 1;"SPECTRUM"
 80 PRINT AT 11,2;FLASH 1;"SPECTRUM"
 90 PRINT AT 13,2;INVERSE 1;"SPECTRUM"
100 PRINT AT 15,2;INK 6;BRIGHT 1;"SPECTRUM"
110 PRINT AT 17,2;INK 6;BRIGHT 1;"SPECTRUM"
```

Dans les lignes de 10 à 40 sont disposées les couleurs bleu pour le contour et le fond, et blanc pour les caractères. La ligne 50 imprime sans rien spécifier (à part les coordonnées) et donc l'écrit sortira blanc sur bleu. La ligne 60 spécifie au contraire la couleur 4(vert) pour les caractères et le message sera donc vert sur bleu. La ligne 70 spécifie que le message doit être à luminosité extra, en respectant toutefois les couleurs établies au début du programme. La ligne suivante, au contraire, imprime le message clignotant, tandis que la ligne 90 l'imprime en champ inversé, c'est-à-dire à caractères bleus sur fond blanc. La ligne 100 spécifie la couleur jaune pour les caractères et la luminosité extra. La dernière ligne indique encore la couleur jaune pour les caractères, mais cette fois l'impression est en champ inversé, donc bleu sur jaune et non pas jaune sur bleu.
Espérons que cet exposé vous aura paru suffisamment clair car, dans le prochain numéro, nous nous occuperons de mettre en pratique tout ce que nous avons appris jusqu'à présent, avec quelques problèmes "classiques" de programmation.
N'oubliez surtout pas notre prochain rendez-vous, dans un mois!

MICROPROCESSEURS
2e Partie

Cette fois-ci, nous nous occuperons du fonctionnement interne d'un UP, de son architecture, de l'assembleur et des adressages afin de vous donner les bases nécessaires et utiles pour une meilleure compréhension du langage machine, sujet que nous affronterons dans le prochain article.

2.1 ARCHITECTURE INTERNE D'UNE CPU

Maintenant entrons plus profondément dans la partie interne de la CPU (Unite Centrale). Dans chacune d'elles, il existe toujours des cases de mémoire indépendantes de celles pilotées extérieurement par "l'adresse Bus". Ces emplacements prennent le nom de "registres" et peuvent être utilisés pour différentes fonctions ou peuvent avoir un rôle bien spécifique.
Leurs dimensions (la capacité de mémoriser un certain nombre de bits) peut varier selon les CPU, et dans une seule CPU, selon les registres: les "memory oriented" (6510 par exemple) où les opérations sont presque toujours basées sur des données de la mémoire externe dans les différentes opérations (pilotée par l'Adresse Bus) et dans laquelle il n'existe pas de grands registres; et les "register oriented" (Z-80), qui possèdent un vaste jeu de registres internes qui prévoit de remplacer la mémoire externe dans les différentes opérations. De toute façon, indépendamment du fait d'appartenir à une famille plutôt qu'à une autre, il existe certains types de registres communs à presque toutes les familles:

* le "PROGRAM COUNTER (PC), un registre dont le seul travail est de garder mémorisée la valeur de l'adresse de mémoire avec laquelle la "donnée Bus" est en communication. Pratiquement, le contenu de ce registre est celui transmis sur "l'Adress Bus" pour sélectionner une donnée case de mémoire;
* le "STACK POINTER" (SP), dont le contenu constitue l'adresse de plusieurs cases de mémoire à utilisation exclusive de la CPU;
* le "STATUS REGISTER" (SR), dont le contenu est une indication du résultat d'une opération exécutée par la CPU.



De ces registres, qui ont uniquement les rôles cités, nous parlerons plus soigneusement plus loin. Un autre registre commun à presque toutes les CPU est:
l'"ACCUMULATEUR" (A) à l'intérieur duquel sont exécutées des opérations logiques, arithmétiques et d'autres instructions spéciales.
Il existe en outre des registres qui peuvent assumer différents rôles et dont le nom varie selon la CPU à laquelle ils appartiennent.

2.2 ARCHITECTURE INTERNE DU Z-80

Z-80 est un type de CPU "register oriented" et c'est une version plus puissante d'une autre fameuse CPU, la 8080.
Intérieurement, elle est formée d'un Program Counter et d'un Stack Pointer de 16 bits de dimensions, qui sont donc capables de se rapporter à une case de mémoire dans un champ de 64K (65536 octets).
L'Accumulateur a la dimension d'un octet, cette CPU étant pourvue d'une Données Bus à 8 bits et le Status Register étant lui-aussi de cette dimension. En plus de ceux-ci, il existe aussi 20 autres registres qui peuvent assumer des rôles différents:
les registres B,C,D,E,H,L et leur jeu alternatif B',C',D',E',H',L'.
Ce sont toujours des registres à 8 bits, à l'intérieur desquels il est possible de mémoriser des données et d'exécuter un certain nombre d'opérations. En plus, ces registres peuvent, en s'accouplant, en former de nouveaux: BC, DE, HL et BC',DE',HL'. Ces registres étant formés de deux de un octet chacun, ils assument une dimension de 16 bits et il est donc possible d'y mémoriser des nombres beaucoup plus grands et d'y exécuter davantage d'opérations.
Les registres IX et IY, appelés registres d'indice, ont chacun une dimension de 16 bits et ont une fonction particulière dont nous parlerons par la suite.
Tous les registres nommés, ainsi que A' et F' (le jeu alternatif de A et de B) forment l'ensemble des registres à adresse générale d'un Z-80.
Il existe en outre, deux autres registres avec des fonctions particulières: l'"INTERRUPT VECTOR REGISTRE (1)" qui exécute des fonctions spéciales lorsque la CPU est interrompue pendant son élaboration par un dispositif externe.
Le "MEMORY REFRESH REGISTER"(R), que l'opérateur ne peut utiliser, connecté à la partie hard d'un ordinateur.
Nous pouvons, à présent, donner une image de l'organisation interne des registres d'une CPU Z-80.

registres à 8 bits

A F	A' F'	I R

B C	B' C'	registres utilisation générale
D E	D' E'	
H L	H' L'	
groupe primaire registres	groupe secondaire registres	

registres à 16 bits

IX	SP
IY	PC

D'après cette figure, nous pouvons relever que le Z-80 possède un jeu d'au moins 22 registres internes, dont 20 sont utilisables par l'opérateur (les seuls non accessibles sont le PC et le R), ce qui permet une puissance et un étirement de programmation du plus haut intérêt.

2.3 CODES MNEMONIQUES

Nous avons vu dans le paragraphe 1.5 qu'une CPU reçoit une instruction sous forme de code numérique, représenté en général sous notation hexadécimale, et que chaque code peut représenter indiféremment une donnée sur laquelle travailler ou une exécution à exécuter!
A ce point, il serait impossible pour un être humain de se rappeler tous les codes correspondants aux instructions à utiliser dans un programme (le Z-80 en possède plus de 600).



Afin de rendre plus simple et plus immédiate la programmation, on emploie les codes mnémoniques, ainsi nommés justement parce que plus faciles à comprendre et à se rappeler. Ces codes sont de véritables mots qui représentent à l'opérateur, de façon rapide, la fonction de l'instruction à exécuter.
La traduction d'un code mnémonique en code numérique, interprétable par la CPU, est du ressort d'un programme externe appelé "Assembleur" lorsque l'on travaille avec, ou encore on utilise des barèmes déterminés qui montrent la correspondance numérique des codes mnémoniques.
On a tendance à différencier la programmation en langage machine avec des codes numériques, de celle en Assembleur avec des codes mnémoniques, mais le résultat, au fond, est le même: celui de composer une série de codes qui formeront un programme en langage machine.

2.4 CODES MNEMONIQUES ZILOG

Les codes mnémoniques qui nous intéressent sont ceux de standard Zilog, la firme qui a projeté le Z-80. Nous tenons cependant à préciser que beaucoup d'assembleurs reconnaissent le standard "Intel", mais de toute façon ces deux standards sont pour ainsi dire identiques. Avant tout, dans une instruction en Assembleur Zilog, on reconnaît toujours un "mnemonic Code" (MN Code), une "Destination Address" (D.A.) une "Source Address" (S.A.) et une "Operation Code" (Op code). L'Op code représente le code numérique correspondant à l'instruction en code mnémonique et c'est la fonction de l'assembleur lui-même de le calculer après avoir introduit le code mnémonique. Le MN code est toujours présent dans une instruction, puisqu'il identifie le genre d'instructions à exécuter. Par exemple, l'instruction avec Op code 76 dont nous avons parlé dans le paragraphe 1.5, a, en assembleur, la représentation suivante [HALT]:

Op code	MN code	D.A.	S.A.
76_{16}	HALT		

Donc, lorsque vous utiliserez un programme assembleur, dans la liste d'instructions à exécuter, vous n'écrirez pas l'Op code 76, mais le code mnémonique HALT.
L'instruction que nous avons à peine introduite ne demande ni de Source Address, ni de Destination Address. Ces dernières, en effet, spécifient respectivement à la CPU où prendre la donnée à employer dans l'opération à exécuter et où mettre le résultat consécutif à l'instruction exécutée. Si nous voulons augmenter

de 1 le contenu numérique de l'accumulateur, l'instruction devient [INCA].

Op code	MN code	D.A.	S.A.
$3C_{16}$	INC	A	

Nous avons, dans ce cas, spécifié dans l'instruction une Destination Address (A), puisque nous voulons que le résultat de l'augmentation soit placé dans l'accumulateur. Nous avons une Source Code si, par exemple, nous voulons transférer une donnée d'un registre (nous utilisons le registre B) à l'accumulateur. La nouvelle instruction devient [LD A,B].

Op code	MN code	D.A.	S.A.
78_{16}	LD	A	B

"LD" spécifie "LOAD" (charger), A spécifie Accumulateur et B le registre B. A ce point l'instruction se lit "chargez A avec le contenu de B". La S. Address indique à la CPU où prendre la donnée à utiliser pour le chargement, tandis que la D. Address indique où cette donnée doit être chargée. La virgule s'emploie seulement comme séparation entre la D. Address et n'a aucun rôle dans l'instruction même.

2.5 ADRESSAGES

Après avoir introduit l'instruction mnémonique, nous pouvons remarquer qu'une instruction se rapporte presque toujours à une donnée que la CPU doit prendre, élaborer, et qui doit ensuite mémoriser le résultat.
En langage machine, toutes les données et les instructions sont contenues dans la RAM externe à la CPU, qui accède aux instructions en lisant une série de cases consécutives qui contiennent les différents octets qui représentent des instructions ou des données.
C'est donc un rôle important qu'exerce le PROGRAM COUNTER, qui garde mémorisé l'adresse de l'instruction que la CPU est en train de lire et qui, après chaque lecture d'instructions, est augmenté pour pointer lors de l'instruction à exécuter suivante.
Nous notons donc que pour, l'exécution des instructions, le PC contient toutes les informations nécessaires.
Au contraire, il sera bien différent de spécifier une position de mémoire sur laquelle exécuter une opération car si, par exemple, nous voulons charger une donnée numérique dans un registre, le PC prévoit de faire lire l'instruction de chargement, tandis que la case de mémoire d'où on chargera doit être spécifiée dans l'ins-



truction même. Il existe donc différente modes de se reporter à une donnée sur laquelle travailler: l'ensemble de ces modes est appelé "Adressage".

2.6 LES MODES D'ADDRESSING

Addressing est un terme anglais pour définir "adressage", c'est un mot commun à toute la littérature concernant CPU et ordinateurs. En particulier, il existe dans une CPU Z-80 différents modes "d'addressing".

REGISTER ADDRESSING (adressages avec registres)

Ce genre d'adressage est celui que nous avons utilisé dans l'instruction précédente (LD A,C) et qui détermine S.A. et D.A., deux registres de la CPU dont les adresses sont insérées dans l'octet même de l'Op code.

IMMEDIATE ADDRESSING (Adressage immédiat)

Une instruction qui utilise ce genre d'adressage a un code formé de deux octets. Par exemple, on obtient l'"immediate addressing" dans l'instruction suivante, dans laquelle nous chargeons le registre C avec le nombre 03H (H = hexadécimal), [LDC,03H[.

Op code	MN code	D.A.	S.A.
$0E_{16}$ 03_{16}	LD	C	03H

Nous notons que cette instruction est formée de deux octets: le premier (0E) spécifie l'instruction de LoaD, tandis que le second (03) est la donnée à charger. Ce genre d'adressage est appelé "immédiat" car l'octet sur lequel travailler suit immédiatement l'Op de l'instruction.

D'autre part, nous pouvons aussi remarquer comment la CPU est capable de distinguer une donnée d'un Op code: l'Op code informe la CPU que l'octet qui sera trouvé dans l'emplacement de mémoire successif est une donnée et non pas un code d'instruction à exécuter.

IMMEDIATE EXTENDED ADDRESSING (adressage étendu immédiat)

Nous avons déjà fait allusion au fait que certains registres peuvent s'unir pour former un registre à 16 bits unique. Pour introduire une donnée dans un de ces couples de registres (qui sont considérés comme un registre unique), on fait appel à l'"Immediate Extended Addressing" qui permet l'emploie de données à 16 bits. Si nous voulons remplir le registre BC (formé de l'accouplement de B et de C) avec le nombre 1023H nous aurons l'instruction [LD BC, 1023H[.

Op code	MN code	D.A.	S.A.
01_{16} 23_{16} 10_{16}	LD	BC,	1023H

Après l'exécution de cette instruction, nous avons chargé un nombre de 16 bits dans BC en utilisant trois octets. Mais puisque la "Donnée Bus" d'un Z-80 est de 8 bits seulement, nous avons décomposé ce nombre en deux octets, en les envoyant en séquence à la CPU derrière le véritable Op code.
Il est a remarquer maintenant, qu'après avoir décomposé 1023H en 10 H et 23 H, nous avons envoyé à la CPU d'abord l'octet de poids faible (23H) et ensuite l'octet de poids fort (10H).
Cette caractéristique est commune à toutes les CPU, et donc, lorsque nous mémorisons un nombre de 16 bits dans deux adresses voisines de 8 bits, l'octet de poids faible est toujours mémorisée dans l'adresse qui est lue en premier, et dans l'adresse suivante l'octet de poids fort. En outre, lorsque nous avons un couple de registres, qui est toujours indiqué par deux lettres voisines placées alphabétiquement (BC,DE,AC), la seconde lettre représente le registre où sera mémorisé l'octet de poids faible. C'est pourquoi, après l'instruction LD BC, 1023H, nous aurons dans le registre C le nombre 23H et dans le registre B le nombre 10H.
EXTENDED ADDRESSING (Adressage étendu)
Une instruction qui utilise l'"Extended Addressing" emploie au moins une adresse à 16 bits, c'est pourquoi, en plus de l'Op code, suivent deux autres octets pour représenter cette adresse. Cet adressage peut être utilisé lorsque nous voulons mémoriser le contenu de l'Accumulateur dans une adresse de mémoire (par exemple 3002H), et donc nous aurons l'instruction [LD (3002H),A].

Op code	MN code	D.A.	S.A.
32_{16} 02_{16} 30_{16}	LD	(3002H),	A

Ici aussi, l'adresse à 16 bits est représentée par deux octets, le premier de poids faible. Les parenthèses entre lesquelles est enfermée l'adresse indiquent que la destination du résultat de l'opération est le "contenu" de la case mémoire indiqué dans l'adresse même, et non le nombre 3002H (une telle instruction n'aurait aucun sens, car on ne peut mémoriser un nombre dans un autre nombre).
REGISTER INDIRECT ADDRESSING (Adressage Indirect avec registre)
Le fonctionnement de cet "addressing" est plus ou moins semblable au précédent puisque, ici aussi, une adresse à 16 bits y est impliquée. La seule différence est que l'adresse ne doit plus être spécifiée comme deux octets consécutifs à l'Op code, mais est contenue à l'intérieur de l'un des registres à 16 bits (BC, DE, HL, SP).
Par exemple, nous pouvons obtenir le même résultat de LD



(3002H), A en remplissant le registre HL avec le nombre 3002H, et indiquer ensuite à la CPU de mémoriser le contenu de A dans l'emplacement indiqué par contenu de HL. Le programme de cette exécution serait donc: LD HL,3002H; remplissez la mémoire indiquée par HL avec le contenu de A Ce qui donne:

Op code	MN code	D.A.	S.A.
77_{16}	LD	(HL)	A

Ici aussi, les parenthèses indiquent comme adresse le contenu du registre HL et non le nombre HL (qui n'existe pas dans le système hexadécimal).
MODIFIED PAGE ADDRESSING (adressage à page zéro modifié)
Cet adressage implique un adressage à 16 bits. Mais seul l'octet de poids faible est fourni à la CPU tandis que l'autre est considéré comme étant toujours égal à 0. C'est pourquoi cet adressage ne peut indiquer que les seuls 256 premiers octets de mémoire (de 0000H à 00FFH). Il est employé avec des instructions particulières dont nous parlerons plus loin.
BIT ADDRESSING (Adressage sur bit)
Le Z-80 a la possibilité d'exécuter des opérations sur un seul bit de mémoire ou de registre. Cet adressage indique sur quel bit une certaine opération doit être exécutée. Les bits sont numérotés de 0 à 7, avec le bit de poid faible (celui plus à droite) numéroté 0 et celui de poids (plus à gauche) 7.
INDEXED ADDRESSING (Adressage indexé)
Dans ce genre d'adressage on utilise les registres indice IX et indice IY.
Pour obtenir l'adresse à employer dans une instruction, il faut additionner au nombre contenu dans l'un de ces deux registres un nombre (offset): La somme donnera l'adresse réelle à utiliser. Par exemple, si nous voulons exécuter l'équivalent de l'instruction LD(3002H), avec l'adressage indexé, nous pouvons charger dans le registre IX le nombre 300H et, par la suite mémoriser A dans l'emplacement 3000H + 02H. Le programme donnerait:

DD_{16} 21_{16} 00_{16} 30_{16}	LD IX, remplissez IX avec le nombre 3000 H
DD_{16} 77_{16} 02_{16}	LD (IX + 02H),A mémorise A dans l'emplacement indiqué

par IX plus deux.
Et donc:

Op code	MN code	D.A.	S.A.
DD_{16} 77_{16} 02_{16}	LD	(IX + 02H) ,	A

Ici aussi les parenthèses spécifient d'employer l'adresse indiquée par la somme de IX et de Ø2H. A noter qu'après l'exécution de cette instruction, IX reste inchangé car la CPU exécute la somme sans interférer avec un registre.
RELATIVE ADDRESSING (adressage relatif)
Ce genre d'adressage est très particulier et nous en parlerons dans la partie concernant les jumps et les branches.
IMPLIED ADDRESSING (Adressage implicite)
Ce genre d'adressage s'emploie fréquemment avec des opérations mathématiques, car celles-ci sont possibles uniquement en utilisant l'Accumulateur en tant que Destination Address. Il est donc implicite dans l'instruction où doit être mis le résultat de l'opération. Par exemple [ADD A,B].

Op code	MN code	D.A.	S.A.
$8Ø_{16}$	ADD	A	B

Cette instruction spécifie d'additionner le contenu du registre B à celui de l'Accumulateur. Il est évident que la destination de ce résultat sera l'Accumulateur lui-même.
Terminons cette seconde partie en espérant avoir été suffisamment clairs. Lors de notre prochain rendez-vous, nous parlerons des véritables instructions du langage machine. Si vous avez déjà appris les "modes d'adressages", ces instructions vous apparaîtront beaucoup plus simples!

INSTRUCTIONS POUR LA CASSETTE DE SOFTHEQUE n° 6

INSTRUCTIONS DE CHARGEMENT

Côté ZX SPECTRUM

Pour charger les programmes introduisez la cassette dans le magnétophone et écrivez "LOAD" (s'obtiennent avec SYMBOL SHIFT et P) immédiatement après appuyez sur la touche ENTER et faites marcher le magnétophone avec PLAY.

Ensuite il suffira de suivre les instructions qui apparaitront sur vidéo. Lorsque vous aurez fini d'utiliser un programme éteignez l'ordinateur en détachant pendant un instant le câble d'alimentation. Tapez encore une fois sur LOAD suivi de ENTER afin de charger le programme suivant.

CHEWIMG-GUM

Lorsque votre programme aura été chargé, les instructions suivantes apparaîtront:

Appuyez sur la touche 1 pour commencer.

Pour déplacer à droite ou à gauche votre puissant laser, utilisez les curseurs (6 et 7).

Pour tirer sur les ennemis imprudents, appuyez sur la touche 0.

A la fin de la partie, le score obtenu sera visualisé.

Pour écrire votre nom et celui de vos amis qui participent au combat, après chaque lettre choisie, avec les touches 6 et 7; pressez 0 (fire); après quoi, le choix effectué, appuyez sur la touche S pour commencer.

Si vous voulez vous rendre compte de votre position dans la classification, déplacez-vous sur le symbole # et appuyez sur 0.

Amusez-vous bien.

SQUARE

Vous rappelez-vous ce jeu fameux "le cube magique"?

Tellement difficile mais, en même temps, tellement passionnant!

Nous vous le reproposons aujourd'hui dans une version spéciale: SQUARE. Sur l'écran de votre Spectrum apparaîtra un carré formé de 9 cases de deux couleurs différentes, violet et vert: vous devrez, en appuyant sur les touches numériques de 1 à 9, changer les blocs de couleurs jusqu'à obtenir un écran complètement vert ou violet. L'ordinateur vous signalera chaque fois le nombre d'essais que nous avez déjà fait: essayez de gagner avec le minimum d'essais possible!

ROULETTE

Peuvent jouer de 1 à 4 personnes; après chaque tour, chacun peut abandonner s'il le veut, tandis que celui qui a épuisé la poule sort automatiquement du jeu.

Les places laissées vacantes peuvent être occupées par d'autres.

Chaque joueur commence avec trois fiches de 100 (colonne à gauche), aucune fiche de 10 et de 1 (colonne au centre et colonne à droite).

Après chaque tour, chacun peut effectuer jusqu'à 3 mises.

La mise maximum est 10.

Voici les mises possibles:
MANQUE - IMPAIR - ROUGE - PASSE - PAIR - NOIR - DOUZAINE - COLONNE - NUMERO.

Pour choisir la mise il suffit d'indiquer les 3 premières lettres.

Après DOUZAINE il faut choisir: P(première), M(deuxième), D(dernière).

Après COLONNE indiquer 1,2,3 (première, deuxième et troisième à partir du bas).

Après NUMERO indiquer naturellement celui qui a été choisi (de 1 à 36).

La même mise ne peut être effectuée deux fois au cours du même tour, même par deux joueurs différents. La limite maximum de gain est - 999 -

Si un des joueurs dépasse cette limite, la partie se termine.

Amusez-vous bien!

GUITARE

Chers amis, si vous avez chez vous une guitare, mais, hélas, ne savez pas en jouer, ce programme est fait pour vous!

Il vous montrera les schémas des principaux accords sans prétendre à vous apprendre à jouer comme Segovia ou comme Van Halen.

Le but de ce programme est de vous permettre de "gratter" la guitare suffisamment pour vous distraire au cours d'une sortie, ou avec des amis, etc.

Nous commencerons par quelques notions de base.

Avant tout, une précision: par commodité, dans le programme, les notes sont indiquées selon la méthode anglaise et américaine, c'est-à-dire avec les sept premières lettres de l'alphabet.

A = LA, B = SI, C = DO, D = RE, E = MI, F = FA, E = SOL

Examinons à présent le manche de la guitare:

La flèche indique la première touche.

Les cordes, sur ce schéma, sont apparemment disposées en sens contraire mais sur toutes les partitions elles sont indiquées de la même manière, c'est-à-dire: la première corde "E" (MI chanterelle) est celle plus en haut sur le schéma, tandis que sur votre

guitare c'est celle plus en bas (si vous ne l'avez pas prise dans le mauvais sens!).

Maintenant accordons la guitare:

En appuyant sur les touches de 1 à B vous entendrez la note qui correspond à la corde et le nom de celle-ci sera indiqué en-dessous. Pour terminer appuyez sur "F".

A présent vous êtes pour ainsi dire prêts à utiliser le programme qui fonctionne de la manière suivante.

Apparaîtra la question: "quelle option? (A,G,S)?"

Les trois lettres indiquent respectivement: voir le schéma d'un accord, voir une liste des premiers tours harmoniques employés dans les chansons, fin du programme.

Si vous répondez A, on vous demandera "quel accord?" Vous pouvez redemander l'accord en une tonalité majeure (EX:"A"), mineure (ex:"A-"), dièse majeur (ex:"A "), dièse mineur (ex:"A -").

On vous montrera ensuite la position des doigts sur le manche pour obtenir cet accord, les doigts sont énumérés de 1 à 4:

1 = index, 2 = majeur, 3 = annulaire, 4 = auriculaire.

Si, au lieu du numéro vous trouverez les deux symboles reliés par un tiret, cela signifie que l'index presse en même temps toutes les cordes d'un signe 1 à l'autre (barres).

Je vous rappelle que les cordes sont pincées uniquement du bout des doigts, au début vous souffrirez un peu, mais surtout n'abandonnez pas, ça en vaut la peine!

BONNE CHANCE!

Appuyez sur le numéro de la corde qui vous intéresse, ou sur "F" (fin).

INSTRUCTIONS DE CHARGEMENT

Côté VIC 20

Prendre la cassette, la mettre dans le magnétophone avec la face choisie, A ou B, vers le haut.

Allumer l'ordinateur, écrire l'instruction 'LOAD' et appuyer sur la touche 'return': l'inscription "PRESS PLAY ON TAPE" apparaîtra; elle signifie "appuyer sur la touche PLAY sur le magnétophone". Exécuter l'ordre et attendre que l'inscription "FOUND INTRODUCTION A" apparaisse sur l'écran; à ce point, appuyer sur la touche d'espacement et attendre quelques secondes, jusqu'à ce que l'écran réapparaisse avec le curseur clignotant.

Si une inscription d'erreur apparaît, ré-enrouler la cassette et recommencer les opérations.

Si tout est O.K., appuyer sur la touche "STOP" du magnétophone, écrire sur l'écran "RUN" et taper "RETURN" sur l'ordinateur; après quelques instants apparaît la première page de notre revue avec la liste des programmes contenus dans la cassette. Lorsque apparaît l'inscription "ENFONCER LA TOUCHE", appuyer sur une touche du clavier au hasard. Dès que l'écran

devient bleu, avec le bord bleu-clair et les inscriptions dans la partie supérieure que vous voyez normalement en allumant l'ordinateur , écrivez de nouveau "LOAD" et appuyez sur "RETURN"; appuyez sur "PLAY" sur le magnétophone et attendez que l'ordinateur trouve le programme suivant, appuyez sur la touche d'espacement et attendez un instant jusqu'à ce que le curseur clignotant réapparaisse. Si aucune inscription d'erreur n'apparaît, appuyez sur "STOP" sur le magnétophone, formez "RUN" sur l'écran et appuyez sur "RETURN" sur l'ordinateur, afin de faire démarrer le programme. Suivez ces simples instructions chaque fois que vous voulez charger un programme du softhèque ordinateur.

Si vous aviez du mal à charger les programmes et si des inscriptions d'erreur (ex. PRINT LOAD ERROR) apparaissaient sur l'écran, vous pourriez essayer de modifier la position des deux têtes à l'aide d'un tournevis introduit dans le trou situé dans la partie supérieure du magnétophone.

Après quelques essais, vous ne devriez plus avoir de problèmes.

Si vous voulez changer de programme, vous avez une alternative: A) appuyez sur la touche: RUN/STOP; si rien ne se produit, en la maintenant toujours enfoncée, tapez une ou plusieurs fois sur la touche "RESTORE": l'écran devrait se vider et redevenir bleu avec le bord bleu clair. Alors écrivez "LOAD" et suivez les instructions habituelles. B) si vous n'arrivez vraiment pas à vous en sortir, n'ayez crainte, éteignez l'ordinateur et réallumez-le, et recommencez en suivant les instructions habituelles pour charger les programmes.

SONAR CAVE

Vous êtes un explorateur qui, par malheur, s'est perdu dans les entrailles de la terre. Pour reconquérir votre liberté, vous devrez avancer à tâtons, caverne après caverne, en vous servant uniquement d'un sonar, dont la note devient de plus en plus aiguë à mesure que vous vous rapprochez de l'une des sorties dissimulées dans la roche. Tandis que vous poursuivez votre recherche, vous devrez faire très attention aux dangereuses chauves-souris qui se jetteront sur vous pour vous tuer de leur morsure venimeuse: pour vous défendre vous avez à votre disposition 3 mines que vous devrez faire exploser en utilisant la touche ←.

Pour vous déplacer dans le noir, utilisez les touches:
@ pour marcher vers le haut,
/ pour vous diriger vers le bas,
= à droite
/ à gauche.

Le score sera calculé sur la base du temps employé pour trouver une sortie (maximum 30 secondes), au tableau atteint et au nombre de mines non encore explosées. Après chaque porte

découverte, on vous accordera de nouvelles possibilités de salut.
A présent, la seule chose qui nous reste à faire est de vous souhaiter...
Bonne chance, explorateur!

LA DERNIERE BATAILLE

Préparez-vous à combattre jusquà la mort! Vos ennemis essaierons de vous toucher, mais ils ne pourront effectuer leur attaque finale que lorsque vous aurez complètement épuisé vos ressources d'énergie.
Défendez-vous donc tant que vous possédez toutes vos forces!
En suivant les instructions et en appuyant sur une touche, les indications suivantes appraîtront:
5 tirez vers le haut
7 tirez vers le bas
z déplacez-vous vers la gauche
y déplacez-vous vers la droite
Pour finir, sur votre écran apparaîtra le champ de bataille où vous serez protégé par une barrière énergétique. Essayez de toucher vos ennemis pour obtenir un score élevé en conduisant une attaque éclair.
En haut, sur l'écran, vous pourrez contrôler les points effectués et en même temps les forces qui vous restent avant de devenir dangeureusement vulnérable.
Lorsque votre bataille sera terminée, le résultat obtenu apparaîtra, ainsi que le score à atteindre.
Bonne chance!

REFLEXOMETRE

Voulez-vous connaître exactement la qualité de vos réflexes?
Si oui, essayez immédiatement ce nouveau programme de votre Vic 20.
Sur l'écran apparaîtra 5 fois de suite "maintenant"! Vous devrez alors appuyer le plus vite possible sur la touche F1. Vous verrez surgir un animal, lièvre, souris, tortue, escargot d'or, escargot d'argent et escargot de bronze, dont chacun représentera un certain degré de rapidité.
A la fin des cinq essais, l'ordinateur vous informera du résultat obtenu, résultat défini par la moyenne de tous les résultats.
Faites attention: n'appuyez pas trop tôt sur F1 sinon votre Vic 20 vous conseillera de retrouver votre calme avant de continuer!
Ce jeu exige à la fois rapidité et contrôle de soi. Ne vous découragez pas après les premiers essais. Entraînez-vous régulièrement et vous verrez qu'en très peu de temps vous parviendrez, vous aussi, à devenir un "lièvre" super-rapide!

PROGRAMMES RECORDER

C'est un programme d'utilité fondamentale, pour créer un fichier de consultation facile. Il permet une économie de temps importante dans la recherche d'un programme spécifique parmi tous ceux que vous possédez déjà. Vous pourrez ainsi les classer selon vos exigences et, éventuellement, en obtenir une liste écrite: vous pourrez gérer des listes de plus de 100 programmes et les enregistrer aussi partiellement, interrompre momentanément l'enregistrement pour le reprendre ultérieurement.

Chaque programme est défini par un nombre progressif, par un nom que vous devrez choisir en utilisant 22 caractères au maximum, par une description qui ne devra pas dépasser 70 caractères et par un code inférieur à 12 caractères. Dans cette dernière rubrique sont incluses toutes les données relatives au langage utilisé, à la longueur en octets, à la position sur bande, etc. Par exemple: le code BA 7250 N001 indiquera le programme en BASIC, long de 7250 octets et repérable sur bande numéro 001.

Les instructions présentes le rendent extrêmement facile à utiliser.

Après le RUN, l'inscription suivante apparaîtra:

1) Création Liste.
2) Lecture Liste.

Pour créer votre fichier, appuyez sur la touche 1: l'ordinateur signalera le numéro du programme et vous demandera les renseignements nécessaires pour le reconnaître. Grace à ces symboles particuliers vous pourrez demander de nouvelles options:

Enregistrement données
Menu principal
Ordre alphabétique

Dans le menu principal, vous pourrez choisir parmi les rubriques suivantes:

1) Révision totale
2) Révision partielle
3) Edition des données
4) Enregistrement des données
5) Création Liste

avec lesquelles vous pourrez apporter d'éventuelles corrections ou passer à l'enregistrement de votre fichier personnel.

Numéros déjà parus:

SOFTHEQUE N. 1
Pour les possesseurs de ZX SPECTRUM et VIC 20
Dans la revue n° 1: Le pseudocode et la programmation de base - Les touches fonctionnelles et le joystick du Vic 20 - Comment augmenter la vitesse des programmes en Basic - La mémorisation des données du ZX Spectrum - Sinclair et Commodore: les nouveautés.
Dans la cassette n° 1: Côté ZX Spectrum: U.F.O - Wargame - Anatomie - Gestion Magasin.
Côté Vic 20: Quotient Intellectuel - Chasse au Trésor - Enfer 3D - Gestion Magasin.

LOGITHEQUE N. 1
Pour les possesseurs de Commodore 64 et TI 99/4A
Dans la revue n° 1: Le pseudocode et la programmation de base - Les touches fonctionnelles du Commodore 64 - Le Basic c'est quoi au juste? - TI pour le son.
Dans la cassette n° 1: Côté Commodore 64: Wargame - Slalom - Quotient Intellectuel - Budget Familial.
Côté TI 99/4A: Splat - Poker - Puissance 4 - Voyage dans l'Espace.

SOFTHEQUE N. 2
Dans la revue n° 2: Pseudocode: Théorie et application des matrices - Qu'est-ce qu'un langage de programmation - Comment développer un programme.
Dans la cassette n° 2: Côté ZX Spectrum: Tour Laser - Course de Grenouilles - Hélibomber - Régime et calories.
Côté Vic 20: Régime - Calories - Tir à la Cible - Go Moku.

LOGITHEQUE N. 2
Dans la revue n° 2: Pseudocode: Théorie et application des matrices - Qu'est-ce qu'un langage de programmation? - Comment développer un programme?
Dans la cassette n° 2: Côté Commodore 64: Formule 1 - Black Jack - Tennis 3D - Gestion Magasin.
Côté TI 99/4A: Labyrinthe - Slot Machine - Agent Secret - Space War.

SOFTHEQUE N. 3
Dans la revue n° 3: Pseudocode: Instructions read et data - Les secrets du Vic 20 - Cours de Basic.
Dans la cassette: Côté ZX Spectrum: Mixxil - Sequencer - Biorythme - Ping Pong.
Côté Vic 20: Attaque Aérienne - Stop Music - Memory Trainer - Change.

LOGITHEQUE N. 3
Dans la revue n° 3: Pseudocode: Instructions read et data - Périphériques d'entrée et sortie - Cours de Basic - Améliorons nos programmes sur le TI 99/4A.
Dans la cassette: Côté Commodore 64: Starway - Poker aux Dés - Régime - Calories.
Côté TI 99/4A: La Goulue - Comptabilité - Les Serpents - Le Coffre - Fort.

SOFTHEQUE N. 4
Dans la revue n° 4: Pseudocode: Structures données complexes - Queue - Liste - Simulation. Les systèmes de numération - Le Basic du Spectrum - Variables internes et Gestion Vidéo.
Dans la cassette: Côté ZX Spectrum: Bilan Familial - Les Gloutons - Ville Souterraine - Le Cerveau.
Côté Vic 20: Vic Calc - Quinze 3D - Abeilles - Horoscope.

LOGITHEQUE N. 4
Dans la revue n° 4: Pseudocode: Structures données complexes - Queue - Liste - Simulation. Les systèmes de numération - Les secrets du CBM 64 - Ti 99 4/A: Comment améliorer l'output sur l'écran.
Dans la cassette: Côté Commodore 64: Sink - Bois Enchanté - Mister Boa - Perspective.
Côté TI 99/4A: Lettres Assassines - Au Secours - Moyenne Scolaire - La Carte plus Forte.

SOFTHEQUE N. 5
Dans la revue: Pseudocode: Nombres Aléatoires - La Distribution des cartes - Le Langage Machine - Le Basic du Spectrum.
Dans la cassette: Côté ZX Spectrum: 3D Graph - Trafic Spatial - Monstres - Agenda Téléphonique.
Côté Vic 20: Les Ponts - Puzzle - Easy Word - Seven Eleven.

LOGITHEQUE N. 5
Dans la revue: Pseudocode: Nombres Aléatoires - La Distribution des cartes - Le Langage Machine - Le Basic du CBM 64 - Fonctions principales.
Dans la cassette: Côté Commodore 64: Histogrammes 3D - Mémo 101 - Seven eleven - Puissance 4.
Côte TI 99/4A: Ti Color Ti Sound - Grand Prix F1 - Régime - Seven Eleven.

Vous pouvez vous procurer les numéros déjà parus en envoyant un chèque bancaire ou postal à l'ordre de Promopublications (34 Champs Elysées 75008 Paris) d'un montant de 85 F + 10,70 F pour frais de port soit 95,70 F par numéro demandé.


SOFTHEQUE
ORDINATEUR
Vidéo-jeux et
programmes pour
ZX SPECTRUM
et VIC 20
6
ZX SPECTRUM
• CHEWING-GUM
• SQUARE
• ROULETTE
• GUITARE FACILE
VIC 20
• SONAR CAVE
• LA DERNIERE
BATAILLE
• REFLEXOMETRE
• PROGRAMMES
RECORDER